Objectives
- Encryption
- `str` methods

Translating to/from ASCII
- Translate a character into its ASCII numeric code using `built-in function ord`
  - `ord('a')` => 97
- Translate an ASCII numeric code into its character using `built-in function chr`
  - `chr(97)` => 'a'

Encryption
- Process of encoding information to keep it secure
- One technique: Substitution Cipher
  - Each character in message is replaced by a new character

Caesar Cipher
- Replace with a character X places away
  - X is your key
- Julius Caesar used it to communicate with his generals
  - "Wrap around"
- Write program(s) to do this in next lab

<table>
<thead>
<tr>
<th>Message</th>
<th>Key</th>
<th>Encoded Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>5</td>
<td>fuuqj</td>
</tr>
<tr>
<td>zebra</td>
<td>5</td>
<td>eijgwf</td>
</tr>
<tr>
<td>the eagle flies at midnight</td>
<td>-5</td>
<td>ocz zbvgz agdzn vo hdyidbco</td>
</tr>
</tbody>
</table>
Next Lab

• Write an encoding/decoding program
  ➢ Encode a message
  ➢ Give to a friend to decode

USING THE STR API

str Methods

• str is a class or a type
• Methods: available operations to perform on str objects
  ➢ Used slightly differently than functions
  ➢ Provide common functionality
• To see all the methods available for the str class
  ➢ help(str)

str Methods

• Example method: find(substring)
  ➢ Finds the index where substring is in string
  ➢ Returns -1 if substring isn’t found
• To call a method:
  ➢ <string>.methodname([arguments])
    ➢ Example: filename.find(".py")

Common str Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>center(width)</td>
<td>Returns a copy of string centered within the given number of columns</td>
</tr>
<tr>
<td>count(sub[, start [, end]])</td>
<td>Return # of non-overlapping occurrences of substring sub in the string.</td>
</tr>
<tr>
<td>endswith(sub), startswith(sub)</td>
<td>Return True iff string ends with/starts with sub</td>
</tr>
<tr>
<td>find(sub[, start [, end]])</td>
<td>Return first index where substring sub is found</td>
</tr>
<tr>
<td>isalpha(), isdigit(), isspace()</td>
<td>Returns True iff string contains letters/digits/whitespace only</td>
</tr>
<tr>
<td>lower(), upper()</td>
<td>Return a copy of string converted to lowercase/uppercase</td>
</tr>
</tbody>
</table>

Common str Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>replace(old, new[, count])</td>
<td>Returns a copy of string with all occurrences of substring old replaced by substring new. If count given, only replaces first count instances.</td>
</tr>
<tr>
<td>split([sep])</td>
<td>Return a list of the words in the string, using sep as the delimiter string. If sep is not specified or is None, any whitespace string is a separator.</td>
</tr>
<tr>
<td>strip()</td>
<td>Return a copy of the string with the leading and trailing whitespace removed</td>
</tr>
<tr>
<td>join(&lt;sequence&gt;)</td>
<td>Return a string which is the concatenation of the strings in the sequence with the string this is called on as the separator</td>
</tr>
<tr>
<td>swapcase()</td>
<td>Return a copy of the string with uppercase characters converted to lowercase and vice versa.</td>
</tr>
</tbody>
</table>
String Methods vs. Functions
• Functions: all “input” as arguments/parameters
  ➢ Example: `len` is a built-in function
  ➢ Called as `len(string)`
• Methods: “input” are argument/parameters and the string the method was called on
  ➢ Example: `string.upper()`

Using `str` Methods
• Modify our search program to find out if the entered string has the `.py` extension

```python
PYTHON_EXT = ".py"
filename = raw_input("Enter a filename: ")
if filename[-len(PYTHON_EXT):] == PYTHON_EXT:
    # Appropriate output
if PYTHON_EXT in filename:
    # Appropriate output
```

Get the Username
• Given the directory formatted as
  ➢ `dir = "/home/www/users/username/"`
• Get the username out

Using `str` Methods
• Modify `binaryToDecimal.py` to verify that the entered string contains only numbers
  ➢ Keep asking them for a number until the string contains only numbers

Using `str` Methods
• Modify `binaryToDecimal.py` to verify that the entered string contains only numbers
  ➢ Keep asking them for a number until the string contains only numbers
• 2nd modification: How could we make sure that entered string contains only 0s and 1s?
Implementing Wheel of Fortune

- Simplifications: no money, no buying vowels, no keeping track of previous guesses, one player
- Functionality
  - Displaying puzzle appropriately
  - Gets guesses from user
    - Either letters or solve the puzzle
  - Keep track of the number of guesses
  - Displays puzzle with guesses filled in
- Think about ...
  - User input robustness?
  - Any special cases?

Differences between real and simulated game

- Players type in letter rather than say it
  - Case matters
  - What if enter more than one letter

User input verification

- How can we ensure that the user typed only one letter?
- How can we ensure that the user typed a letter?

Checking the guess

- How can we tell if the guessed letter is in the puzzle?
- How can report the number of times the guessed letter occurs in the puzzle?

How many times should we prompt the user for a guess?

How can we display the current puzzle?

- What does the puzzle look like when we start the game?
- What does it look like after we correctly guess a letter?

Practice: Modify displayed puzzle to handle punctuation

- Include punctuation in displayed puzzle
- Original code:

```python
displayedpuzzle = "" for char in PHRASE:
  if char != " ":
    displayedpuzzle += "_"
  else:
    displayedpuzzle += " "
```

Extra Credit Opportunity

- Modify Wheel of Fortune to …
  - Include prize money
  - Spinning wheel
  - Handles buying vowels vs consonants
  - Multiple players, losing turns
- Due Friday, February 27
For Friday

- Read about Excel 2007 bug
- Lab 5